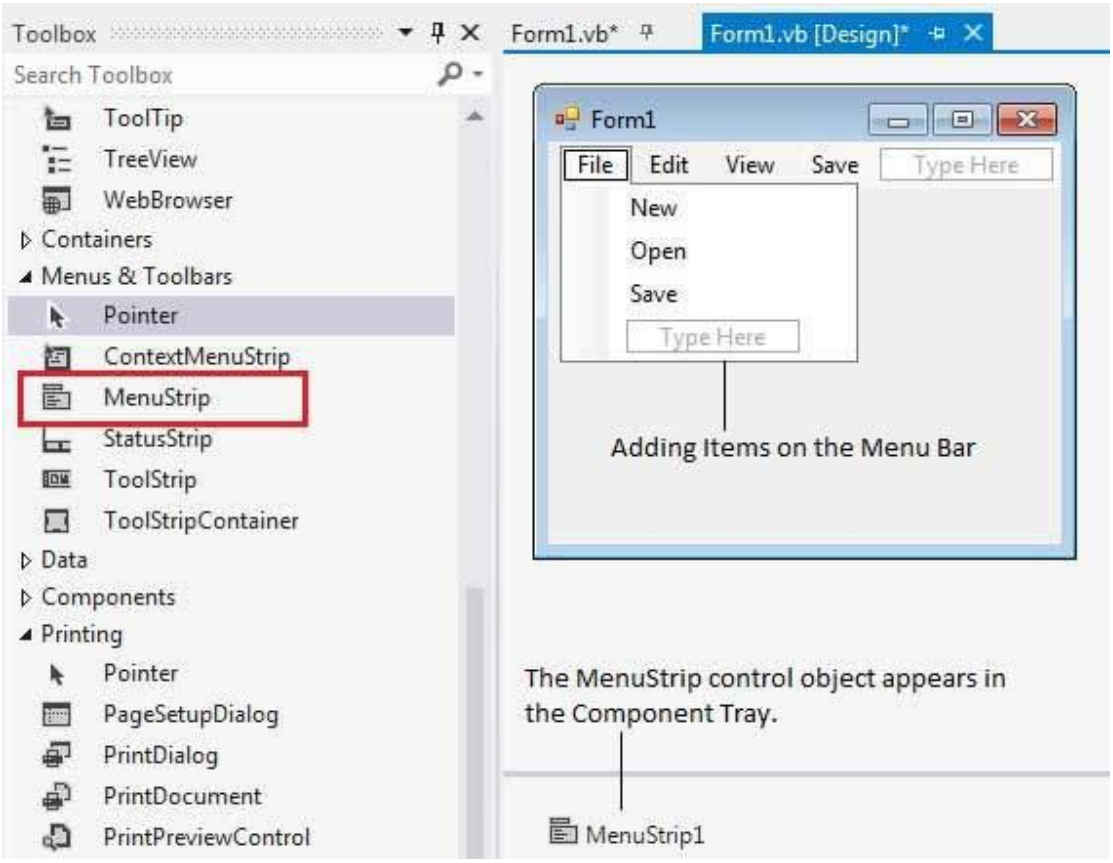
MenuStrip Control

The **MenuStrip** control represents the container for the menu structure.

The MenuStrip control works as the top-level container for the menu structure. The ToolStripMenuItem class and the ToolStripDropDownMenu class provide the functionalities to create menu items, sub menus and drop-down menus.

The following diagram shows adding a MenuStrip control on the form:



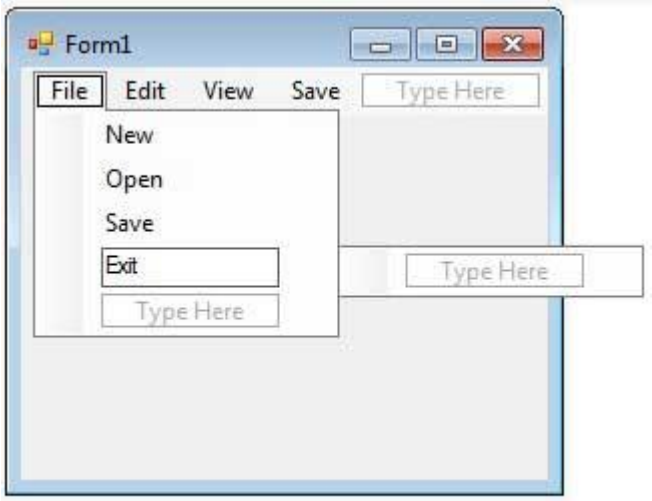
Example

In this example, let us add menu and sub-menu items.

Take the following steps:

- Drag and drop or double click on a MenuStrip control, to add it to the form.
- Click the Type Here text to open a text box and enter the names of the menu items or sub-menu items you want. When you add a sub-menu, another text box with 'Type Here' text opens below it.
- Complete the menu structure shown in the diagram above.

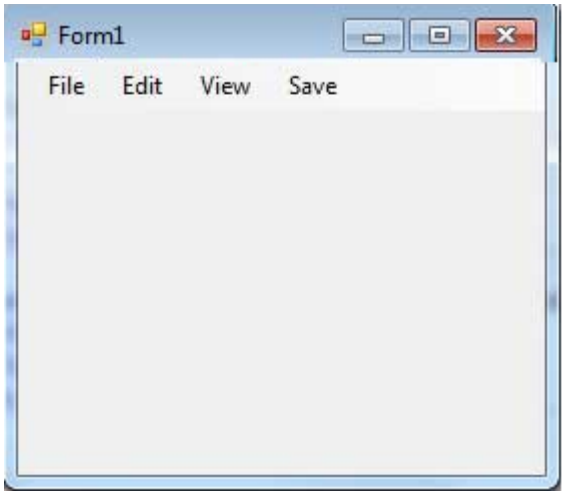
- Add a sub menu **Exit** under the **File** menu.



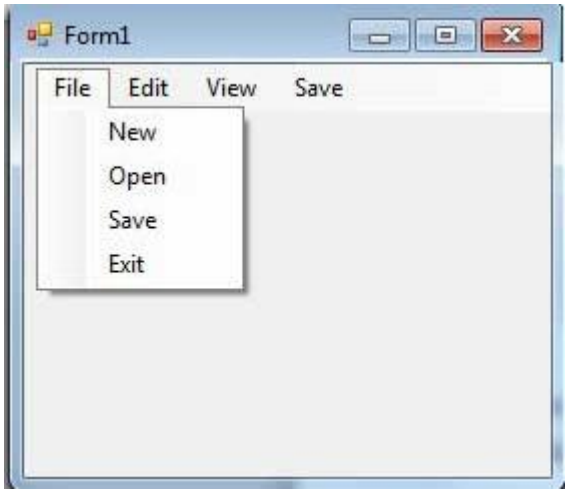
Double-Click the Exit menu created and add the following code to the **Click** event of **ExitToolStripMenuItem**:

```
Private Sub ExitToolStripMenuItem_Click(sender As Object, e As EventArgs) _

   Handles ExitToolStripMenuItem.Click

   End

End Sub
```

When the above code is executed and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window:



Click on the File -> Exit to exit from the application:


Form1
File
Edit
View
Save
New
Open
Save
Exit

ToolStripMenuItem

The **ToolStripMenuItem** class supports the menus and menu items in a menu system. You handle these menu items through the click events in a menu system.

Example

In this example, let us continue with the above example. Let us:

- Hide and display menu items.
- Disable and enable menu items.
- Set access keys for menu items
- Set shortcut keys for menu items.

Hide and Display Menu Items

The **Visible** property of the **ToolStripMenuItem** class allows you to hide or show a menu item. Let us hide the Project Menu on the menu bar.

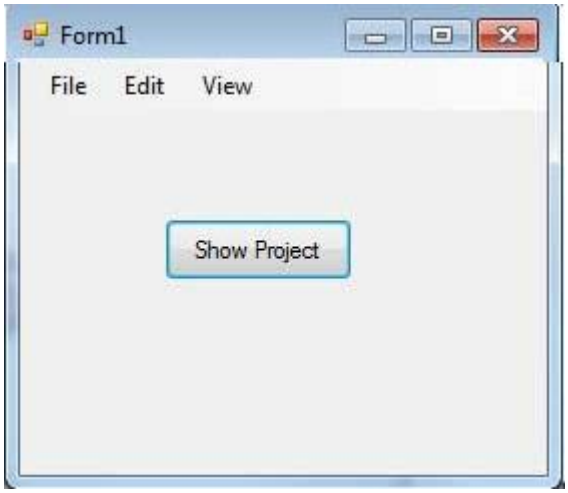
- Add the following code snippet to the Form1_Load event:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) _
Handles MyBase.Load
   ' Hide the project menu
   ProjectToolStripMenuItem1.Visible = False
End Sub
```

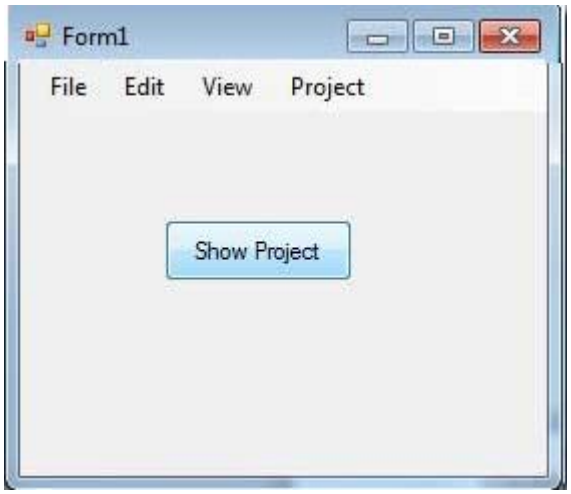
- Add a button control on the form with text 'Show Project'.
- Add the following code snippet to the Button1_Click event:

```
Private Sub Button1_Click(sender As Object, e As EventArgs) _
Handles Button1.Click
   ProjectToolStripMenuItem1.Visible = True
End Sub
```

When the above code is executed and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window:



Clicking on the Show Project button displays the project menu:



Disable and Enable Menu Items

The **Enabled** property allows you to disable or gray out a menu item. Let us disable the Project Menu on the menu bar.

- Add the following code snippet to the Form1_Load event:

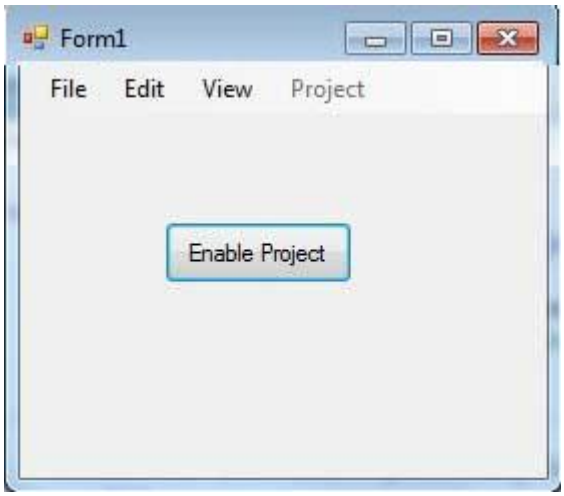
```
Private Sub Form1_Load(sender As Object, e As EventArgs) _
Handles MyBase.Load
   ' Disable the project menu
   ProjectToolStripMenuItem1.Enabled = False
End Sub
```

- Add a button control on the form with text 'Enable Project'.
- Add the following code snippet to the Button1_Click event:

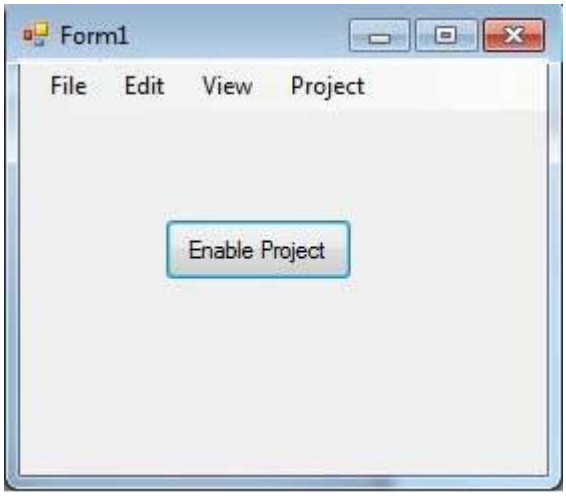
```
Private Sub Button1_Click(sender As Object, e As EventArgs) _
```

```
Handles Button1.Click

   ProjectToolStripMenuItem1.Enabled = True

End Sub
```

When the above code is executed and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window:



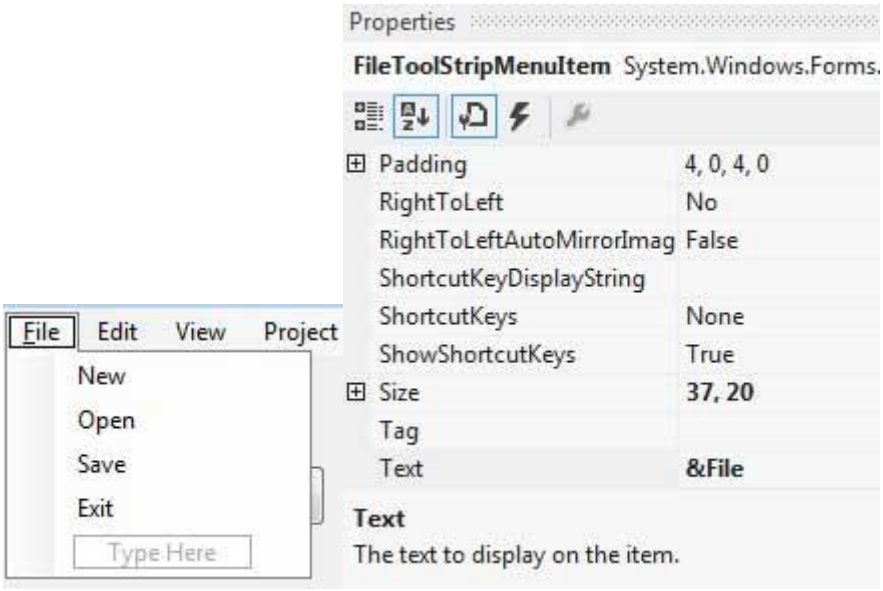
Clicking on the Enable Project button enables the project menu:



Set Access Keys for Menu Items

Setting access keys for a menu allows a user to select it from the keyboard by using the ALT key.

For example, if you want to set an access key ALT + F for the file menu, change its **Text** with an added & (ampersand) preceding the access key letter. In other words, you change the text property of the file menu to &File.

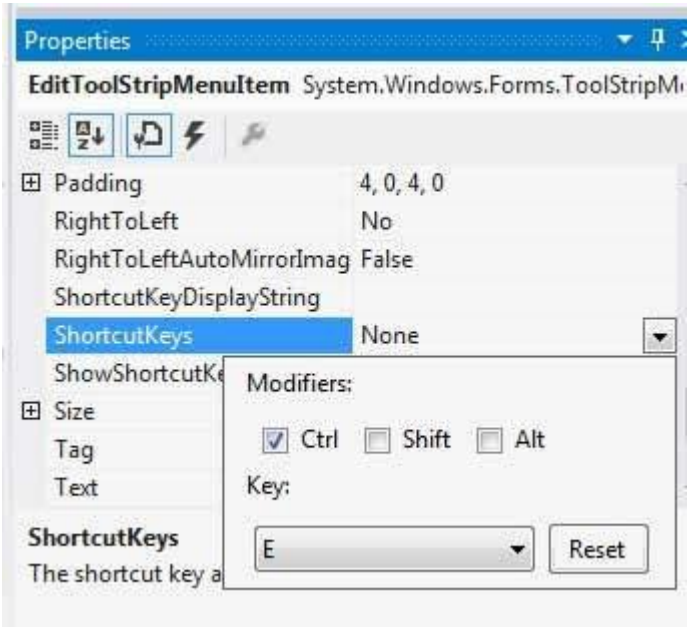


Set Shortcut Keys for Menu Items

When you set a shortcut key for a menu item, user can press the shortcut from the keyboard and it would result in occurrence of the Click event of the menu.

A shortcut key is set for a menu item using the ShortcutKeys property. For example, to set a shortcut key CTRL + E, for the Edit menu:

- Select the Edit menu item and select its ShortcutKeys property in the properties window.
- Click the drop down button next to it.
- Select Ctrl as Modifier and E as the key.



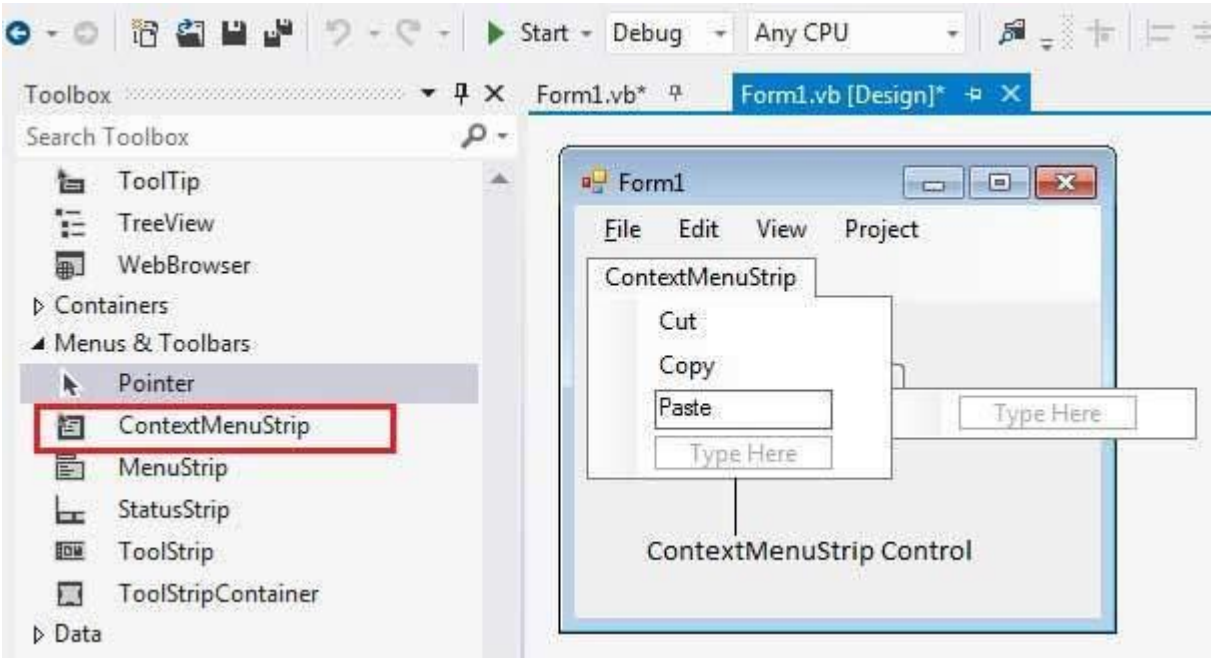
ContextMenuStrip Control

The **ContextMenuStrip** control represents a shortcut menu that pops up over controls, usually when you right click them. They appear in context of some specific controls, so are called context menus. For example, Cut, Copy or Paste options.

This control associates the context menu with other menu items by setting that menu item's ContextMenuStrip property to the ContextMenuStrip control you designed.

Context menu items can also be disabled, hidden or deleted. You can also show a context menu with the help of the Show method of the ContextMenuStrip control.

The following diagram shows adding a ContextMenuStrip control on the form:



Example

In this example, let us add a content menu with the menu items Cut, Copy and Paste.

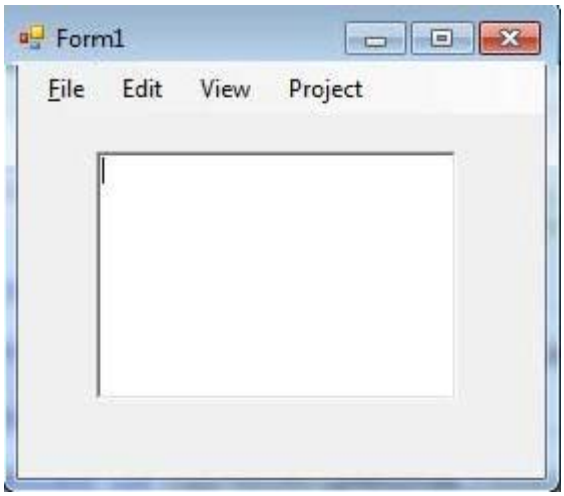
Take the following steps:

- Drag and drop or double click on a ControlMenuStrip control to add it to the form.
- Add the menu items, Cut, Copy and Paste to it.
- Add a RichTextBox control on the form.

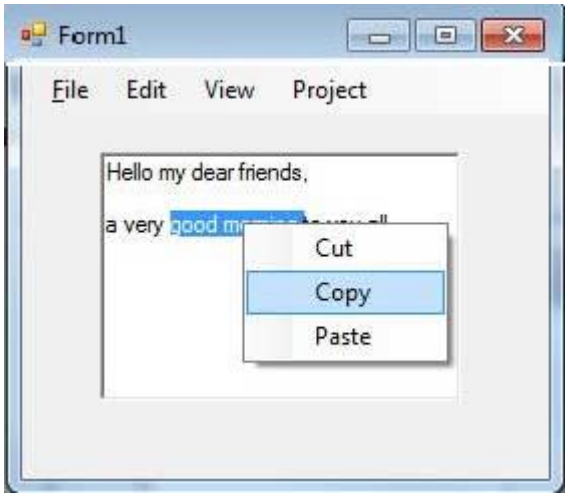
- Set the ContextMenuStrip property of the rich text box to ContextMenuStrip1 using the properties window.
- Double the menu items and add following codes in the Click event of these menus:

```
Private Sub CutToolStripMenuItem_Click(sender As Object, e As EventArgs) _
Handles CutToolStripMenuItem.Click
   RichTextBox1.Cut()
End Sub

Private Sub CopyToolStripMenuItem_Click(sender As Object, e As EventArgs) _
Handles CopyToolStripMenuItem.Click
   RichTextBox1.Copy()
End Sub

Private Sub PasteToolStripMenuItem_Click(sender As Object, e As EventArgs)
_
Handles PasteToolStripMenuItem.Click
   RichTextBox1.Paste()
End Sub
```

When the above code is executed and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window:



Enter some text in the rich text box, select it and right-click to get the context menu appear:



Now, you can select any menu items and perform cut, copy or paste on the text box.

Event Handling

Events are basically a user action like key press, clicks, mouse movements, etc., or some occurrence like system generated notifications. Applications need to respond to events when they occur.

Clicking on a button, or entering some text in a text box, or clicking on a menu item, all are examples of events. An event is an action that calls a function or may cause another event.

Event handlers are functions that tell how to respond to an event.

VB.Net is an event-driven language. There are mainly two types of events:

- Mouse events
- Keyboard events

Handling Mouse Events

Mouse events occur with mouse movements in forms and controls. Following are the various mouse events related with a Control class:

- **MouseDown** - it occurs when a mouse button is pressed
- **MouseEnter** - it occurs when the mouse pointer enters the control
- **MouseHover** - it occurs when the mouse pointer hovers over the control
- **MouseLeave** - it occurs when the mouse pointer leaves the control
- **MouseMove** - it occurs when the mouse pointer moves over the control
- **MouseUp** - it occurs when the mouse pointer is over the control and the mouse button is released
- **MouseWheel** - it occurs when the mouse wheel moves and the control has focus

The event handlers of the mouse events get an argument of type **MouseEventArgs**. The MouseEventArgs object is used for handling mouse events. It has the following properties:

- **Buttons** - indicates the mouse button pressed
- **Clicks** - indicates the number of clicks
- **Delta** - indicates the number of detents the mouse wheel rotated

- **X** - indicates the x-coordinate of mouse click
- **Y** - indicates the y-coordinate of mouse click

Example

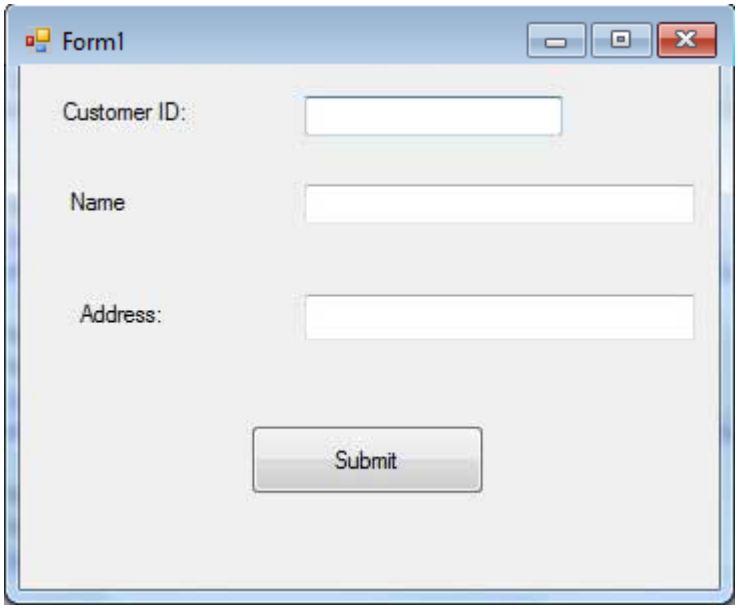
Following is an example, which shows how to handle mouse events. Take the following steps:

- Add three labels, three text boxes and a button control in the form.
- Change the text properties of the labels to - Customer ID, Name and Address, respectively.
- Change the name properties of the text boxes to txtID, txtName and txtAddress, respectively.
- Change the text property of the button to 'Submit'.
- Add the following code in the code editor window:

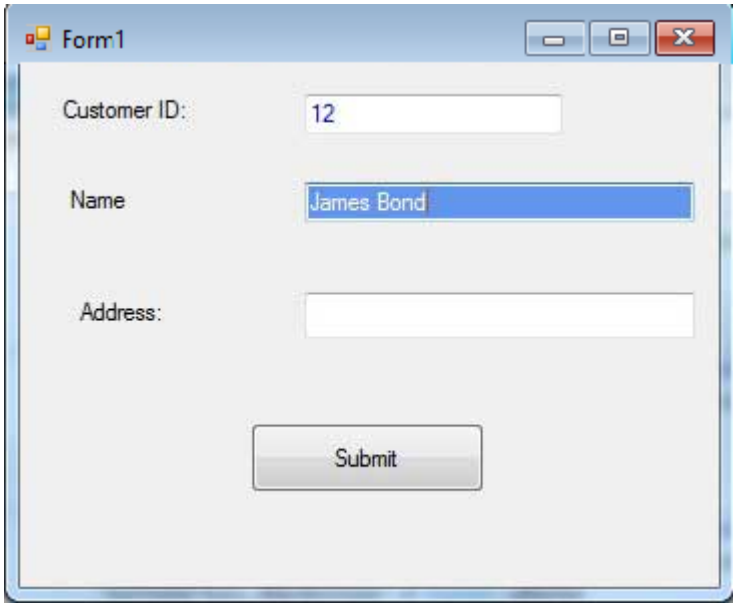
```
Public Class Form1

   Private Sub txtID_MouseEnter(sender As Object, e As EventArgs)_

        Handles txtID.MouseEnter

      'code for handling mouse enter on ID textbox

      txtID.BackColor = Color.CornflowerBlue

      txtID.ForeColor = Color.White

   End Sub

   Private Sub txtID_MouseLeave(sender As Object, e As EventArgs) _

        Handles txtID.MouseLeave

      'code for handling mouse leave on ID textbox

      txtID.BackColor = Color.White

      txtID.ForeColor = Color.Blue

End Sub

   Private Sub txtName_MouseEnter(sender As Object, e As EventArgs) _

       Handles txtName.MouseEnter

      'code for handling mouse enter on Name textbox

      txtName.BackColor = Color.CornflowerBlue

      txtName.ForeColor = Color.White
```

```
    End Sub
    Private Sub txtName_MouseLeave(sender As Object, e As EventArgs) _
       Handles txtName.MouseLeave
       'code for handling mouse leave on Name textbox
       txtName.BackColor = Color.White
       txtName.ForeColor = Color.Blue
    End Sub
    Private Sub txtAddress_MouseEnter(sender As Object, e As EventArgs) _
       Handles txtAddress.MouseEnter
       'code for handling mouse enter on Address textbox
       txtAddress.BackColor = Color.CornflowerBlue
       txtAddress.ForeColor = Color.White
    End Sub
    Private Sub txtAddress_MouseLeave(sender As Object, e As EventArgs) _
         Handles txtAddress.MouseLeave
       'code for handling mouse leave on Address textbox
       txtAddress.BackColor = Color.White
       txtAddress.ForeColor = Color.Blue
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) _
        Handles Button1.Click
       MsgBox("Thank you " & txtName.Text & ", for your kind cooperation")
    End Sub
End Class
```

When the above code is executed and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window:



Try to enter text in the text boxes and check the mouse events:



Reference: https://www.tutorialspoint.com